Shared memory parallelism and low-rank approximation techniques applied to direct solvers in FEM simulation

Patrick Amestoy, Alfredo Buttari, Guillaume Joslin, Jean-Yves L'Excellent, Mohamed Sid-Lakhdar, Clement Weisbecker, Michele Forzan, Cristian Pozza, Rémy Perrin, Valène Pellissier

Shared-Memory Parallelism and Low-Rank Approximation Techniques Applied to Direct Solvers in FEM Simulation

Patrick Amestoy[1], Alfredo Buttari[2], Guillaume Joslin[3], Jean-Yves L'Excellent[4,5], Mohamed Sid-Lakhdar[5], Clément Weisbecker[1], Michele Forzan[6], Cristian Pozza[6], Remy Perrin[7], and Valène Pellissier[7]

[1]Institut National Polytechnique de Toulouse, Institut de Recherche en Informatique de Toulouse, Toulouse 60026, France
[2]Centre national de la recherche scientifique, Institut de Recherche en Informatique de Toulouse, Toulouse 31000, France
[3]Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, Toulouse 31000, France
[4]Institut national de recherche en informatique et en automatique, Montbonnot 38334, France
[5]École Normale Supérieure, Lyon 69364, France
[6]Department of Industrial Engineering, University of Padova, Padova 35131, Italy
[7]CEDRAT, Meylan 38240, France

In this paper, the performance of a parallel sparse direct solver on a shared-memory multicore system is presented. Large size test matrices arising from finite element simulation of induction heating industrial applications are used to evaluate the performance improvements due to low-rank representations and multicore parallelization.

Index Terms— **Eddy currents, FEMs, linear systems, parallel algorithms, sparse matrices.**

I. Introduction

In 3-D finite element simulation of induction heating processes, the solution time is a limiting factor in the design and optimization of new devices. Time-harmonic electromagnetic (EM) problems coupled with thermal (TH) problems are solved in sequence, and the linear system solution in the EM problem is often the bottleneck. For this reason, solver performances have been verified only for the solution of the EM problem with constant material properties in a quasi-stationary time-harmonic regime. Although iterative methods are widely applied for the solution of this kind of problem, in this paper the performances of direct solvers are investigated. Therefore, an efficient direct method to solve large sparse complex matrices should exploit parallelization and reduce memory consumption. In this paper, a reduction of memory requirements through low-rank (LR) techniques and factorization times through shared-memory parallelism in multifrontal massively parallel sparse direct solver (MUMPS) will be discussed [4]. Memory usage and factorization times are compared with another popular sparse direct solver, parallel direct solver (PARDISO) [11]. Matrices arise from the modelization of induction heating industrial devices. Heating of a susceptor by pancake coils and gear induction hardening is taken as test benchmarks [1]–[3]. Geometric model design, meshing, and matrix building are performed by a commercial software [5]. Starting from the same geometry (Pancake or Gear), meshes are gradually refined to solve problems of different sizes leading to matrices ranging from 320 k to 2.5 M degrees of freedom.

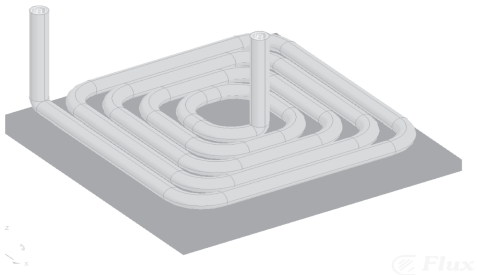

Fig. 1. Geometry of the pancake coil.

II. Description of the First Benchmark

To assess the performance of different sparse solvers for complex linear systems, the model of a pancake inductor designed to heat a graphite susceptor has been taken as the first benchmark problem. This system is used in a furnace for the production of solar grade silicon by the directional solidification system I-DSS [1]. In Fig. 1, the model geometry is presented and Fig. 2 shows its mesh. The numerical solution of the EM problem has been carried out by applying the well known A, AV formulation and using the complex representation of the sinusoidal quantities [2], [14]. The magnetic vector potential coupled with the scalar electric potential formulation has been chosen because it guarantees a precise solution of the eddy current distribution, mostly in models where there is no highly permeable conducting region [14], [15]. Usually this formulation originates systems with more degrees of freedom than the so-called scalar one (used in the second benchmark). This model has been solved using four different types of discretizations, with a number of elements ranging from 600 k up to 2.9 M volume elements, leading to complex sparse systems from 320 k to 1.5 M unknowns. Typical eddy currents distribution is shown in Fig. 3.

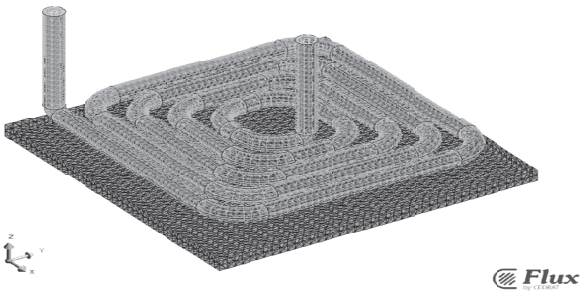

Fig. 2. Mesh of the pancake coil.

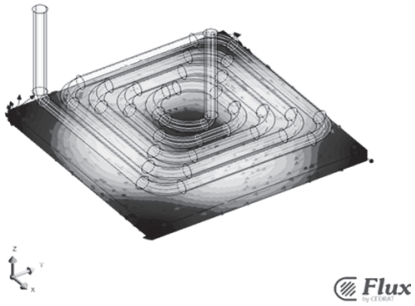

Fig. 3. Eddy current distribution in the graphite susceptor. Values range from 1.2 to 120 kA/m^2.

III. Description of the Second Benchmark

Simulations of induction hardening process is one of the most challenging tasks for multi-physics simulation. A complete process description requires not only to compute the EM-TH coupled problem, but also to couple these results with other different physics, able to calculate the metallurgical transformation and residual stresses [3]. To obtain reliable results of the entire process, the solution of EM-TH coupled problem must be as accurate as possible. To properly describe the EM field distribution, a fine and possibly mapped mesh has to be built mostly in the heat affected region. In general, the model should consider the non-linear material properties of the steel, and consequently the solution requires to apply an iterative method. Furthermore, to properly describe the dependence upon temperature of material properties, the time-dependent TH problem, part of coupled solution, has to be solved using several time steps.

Sometimes, all these requirements lead to a very long computation time. For the purpose of this paper, only the EM steady-state problem is solved with linear magnetic properties. The model used as benchmark describes a slice of the whole system and it is shown in Fig. 4. The 3-D eddy current problem is solved through T-$T0$-ϕ formulation where the magnetic field within conducting regions is

$$H = T + T_0 - \nabla\Phi \tag{1}$$

where T (A/m) is the unknown electric vector potential and ϕ (A) is the total magnetic scalar potential. T_0 (A/m) represents the given imposed electric vector potential, which describes a prescribed current distribution, fed by an external source. Then, the induced current density can be derived from

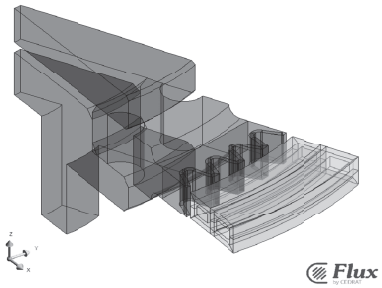

Fig. 4. Figure represents only a slice of the whole inductor workpiece geometry because of symmetries.

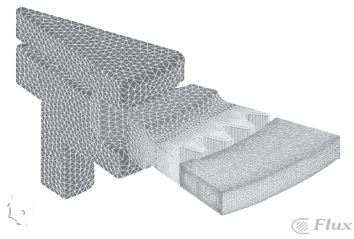

Fig. 5. Mesh of inductor, workpiece, and clampers.

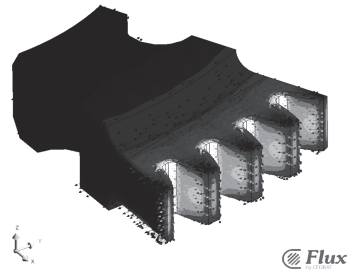

Fig. 6. Eddy current distribution in the gear. Values range from 200 kA/m^2 to 300 MA/m^2.

the Ampère's law [14]

$$J = \nabla \times (T + T_0). \tag{2}$$

The model mesh, shown in Fig. 5, contains 5×10^5 volume elements. With this discretization, the dimension of linear system is 370 k or 2.5 M when, respectively, first- and second-order elements are used. Typical eddy currents distribution is shown in Fig. 6.

IV. Testing Environment

The hardware platform for the evaluation is a dual Intel Xeon (X5670, six cores) with 96 Gb RAM. MUMPS version 4.10.0 [4] and another popular tool Intel MKL parallel direct solver (PARDISO) version 10.3.9 are compared, both in the in-core (IC) and out-of-core (OOC) mode. The effect of swapping temporary data on virtual memory is also evaluated. Configuration details are reported in Table I.

V. Preliminary Results

Although MUMPS is based on a message passing model for parallel computation, a single MPI process with multithreaded

TABLE I
TESTED MATRICES

Test matrix	Pancake 1	Pancake 2	Pancake 3	Pancake 4	Gear 1	Gear 2
Degrees of freedom	320k	630k	1M	1.5M	370k	2.5M
Number of entries in LU factors	990M	2.1G	5.2G	8.7G	700M	12.5G

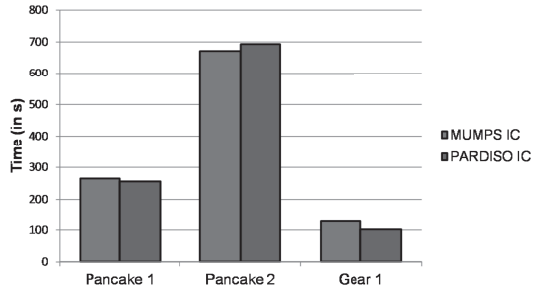

Fig. 7. IC factorization time (12 cores, small cases).

TABLE II
MEMORY NEEDED FOR FACTORIZATION (GB, LARGE CASES)

Test matrix	Pancake 3	Pancake 4	Gear 2
MUMPS OOC	21	30	35
PARDISO OOC	87	Error	Error
MUMPS IC + swap	93	155	218
PARDISO IC + swap	86	147	214

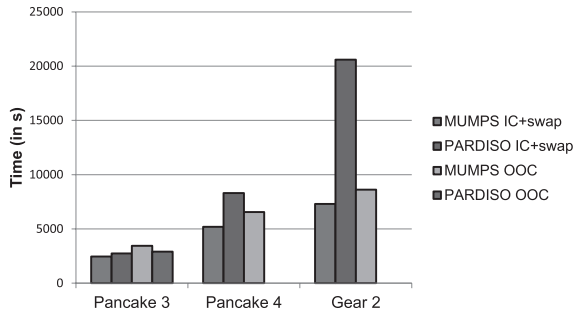

Fig. 8. IC (with swap) and OOC factorization times for large cases.

BLAS libraries is used for tests, as a better performance with such a configuration has been observed on multicore architectures. Both MUMPS and PARDISO are executed using 12 threads and the corresponding factorization times are shown in Fig. 7 and Table II. The IC factorization times are slightly different with MUMPS and PARDISO, leading to very close behavior on 12 cores (see Fig. 7). Memory usage for IC executions is also very similar (see Table II). In the OOC mode, MUMPS is slightly slower than PARDISO on small matrices, but uses less memory since PARDISO does not perform I/O when memory is found to be sufficient, as for Pancake 3.

The missing values for PARDISO in OOC (in Table II and Fig. 8) are due to an error during the factorization step in OOC and with multithreads. Timings on largest problems (Fig. 8) show that MUMPS is less penalized by memory swapping than PARDISO.

TABLE III
LR IMPROVEMENTS

Test matrix	Pancake 2	Pancake 4	
LR threshold ε	10^{-8}	10^{-10}	10^{-14}
Memory saved for factors storage	35%	34%	14%
Operations saved for factorization	60%	60%	29%
Scaled residual	2.3×10^{-16}	1×10^{-12}	3.5×10^{-16}

VI. IMPROVEMENTS BY LR APPROXIMATION TECHNIQUES

A LR matrix can be represented in a form which decreases its memory requirements and the complexity of involved basic linear algebra operations, such as matrix–matrix products. This is formalized in [6].

Let A be a matrix of size $m \times n$. Let k_ε be the approximated numerical rank of A at accuracy ε. A is said to be a LR matrix if there exist three matrices W of size $m \times k_\varepsilon$, Z of size $n \times k_\varepsilon$, and E of size $m \times n$ such that

$$A = W \cdot Z^T + E \tag{3}$$

where $\|E\|^2 \leq \varepsilon$ and $k_\varepsilon\ (m+n) < mn$.

k_ε is commonly referred to as the numerical rank at precision ε and can be computed, together with W and Z with a rank-revealing QR factorization. LR approximation techniques are based upon the idea to ignore E and simply represent A as the product of W and Z^T, at accuracy ε.

In practice, matrices coming from applicative problems are not LR, which means that they cannot be directly approximated. However, LR approximations can be performed on sub-blocks defined by an appropriately chosen partitioning of matrix indexes [6], [7]. Theoretical studies based on mathematical properties of the underlying operators have shown that variable sets that are far away in the domain are likely to have weak interactions which translates into the fact that the corresponding matrix block has a LR. To benefit from this property, a format called block LR (BLR) is used [8]. This format can be exploited within internal data structures of the multifrontal method used in the MUMPS software to decrease the memory consumption and the operation count of the solver [9].

In Table III, results show that the method is more efficient on large problems, which is a good property in the context of large scale computing (note that there is no compression for Pancake 2 with $\varepsilon = 10^{-10}$). This property has also been observed in many other applications. For Pancake 2,

significant gains are obtained with a LR threshold of 10^{-8}; the memory footprint is reduced by almost a factor of three and the complexity is almost halved. In this case, a few steps of iterative refinement are performed to recover full precision from the original approximated precision of 10^{-10}. For Pancake 4 with LR threshold precision of 10^{-14}, a good compression is obtained naturally without any loss of accuracy. Note that by choosing a more aggressive threshold ε, the BLR format can be used to efficiently produce loosely approximated factors that can be used as effective preconditioners for iterative solvers.

VII. Improvements of Shared-Memory Parallelism in MUMPS

To solve a sparse linear system, the multifrontal method transforms the initial sparse matrix into a (elimination) tree of much smaller dense matrices. This tree is traversed in a topological order and each node is computed following a partial LU decomposition. Then, a two-pass solve operation is applied on each node of the tree to find the solution of the original system.

The structure of the tree offers an inner parallelism, called tree parallelism, in which different processes work on data subsets, stored on different nodes of a network. This kind of parallelism is already exploited by MUMPS in distributed memory environments [9]. However, in shared-memory environments only node parallelism is applied; many processes collaborate on the decomposition, working on the same node. In this approach, threaded BLAS libraries are preferred to parallelize dense matrix operations. To make a step further, the fork-join model of parallelism has been implemented in a new code, based on OpenMP programming. Our first goal was then to exploit tree parallelism in shared-memory environments, by adopting algorithms commonly used in distributed memory environments and by rearranging them to a multithreading model. Therefore, the so-called AlgL0 algorithm consists in finding a separating layer in the tree, called L0, such that node parallelism will still be applied above it, but tree parallelism will be applied under it through OpenMP.

The use of adequate memory mapping policies allows to improve the performance of MUMPS on symmetric multiprocessor and non-uniform memory access architectures. The local alloc policy, which consists in mapping the memory pages on the local memory of the processor that first touches them, is applied on data structures used under L0, in order to achieve a better data locality and cache exploitation. The interleave policy, which consists in allocating the memory pages on all memory banks in a round-robin fashion such that the allocated memory is spread over all the physical memory, has been used on data structures above L0 in order to improve the bandwidth.

Both MUMPS 4.10.0 and MUMPS 4.10.0 with new AlgL0 algorithm have been tested on the set of test matrices, with the stated memory allocation techniques. As reported in Table IV, this approach brings a remarkable reduction of computational time on all tested matrices. The benefit tends to decrease on large matrices because the fraction of workload in the top of the tree (above L0) gradually increases in comparison with the workload in the bottom of the tree. Furthermore, this gain should rise through the use of the interleave policy.

TABLE IV
Time Save by Using MUMPS 4.10.0 With AlgL0 Algorithm

Test matrix	Pancake 1	Pancake 2	Pancake 3	Gear 1
Time saving	13%	11%	8%	21%

VIII. Conclusion

Significant improvements of parallel sparse direct solver MUMPS have been successfully tested in 3-D numerical simulation of induction heating industrial applications. Results show a remarkable reduction of both memory usage and number of performed operations. Furthermore, both tree and node parallelism are exploited to reduce the solution time.

References

[1] F. Dughiero, M. Forzan, D. Ciscato, and F. Giusto, "Multi-crystalline silicon ingots growth with an innovative induction heating directional solidification furnace," in *Proc. 37th IEEE PVSC*, Jun. 2011, pp. 002151–002156.

[2] F. Dughiero, M. Forzan, C. Pozza, and E. Sieni, "A translational coupled electromagnetic and thermal innovative model for induction welding of tubes," *IEEE Trans. Magn.*, vol. 48, no. 2, pp. 483–486, Feb. 2012.

[3] S. Lupi, F. Dughiero, and M. Forzan, "Modelling single- and double-frequency induction hardening of gear-wheels," in *Proc. 5th Int. Symp. Electromagn. Process. Mater.*, Sendai, Japan, 2006, pp. 473–478.

[4] (2013, Sep.). *MUMPS* [Online]. Available: http://graal.ens-lyon.fr/MUMPS/

[5] (2013, Sep.). *CEDRAT* [Online]. Available: http://www.cedrat.com

[6] M. Bebendorf, *Hierarchical Matrices—A Means to Efficiently Solve Elliptic Boundary Value Problems* (Lecture Notes in Computational Science and Engineering), 1st ed. New York, NY, USA: Springer-Verlag, 2008.

[7] S. Börm, *Efficient Numerical Methods for Non-Local Operators*. Zürich, Switzerland: European Mathematical Society, 2010.

[8] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker, "Improving multifrontal methods by means of low-rank approximations techniques," in *Proc. SIAM Conf. Appl. Linear Algebra*, Valencia, Spain, 2012, pp. 1–25.

[9] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker, "Grouping variables in frontal matrices to improve low-rank approximations in a multifrontal solver," in *Proc. Int. Conf. Precond. Tech. Sci. Ind. Appl.*, Bordeaux, France, 2011, pp. 1–63.

[10] P. Amestoy, A. Buttari, A. Guermouche, J.-Y. L'Excellent, and M. Sid-Lakhdar, "Exploiting multithreaded tree parallelism for multicore systems in a parallel multifrontal solver," in *Proc. 15th SIAM Conf. Parallel Process. Sci. Comput.*, Savannah, GA, USA, 2012, pp. 1–25.

[11] O. Schenk and K. Gärtner, "Solving unsymmetric sparse systems of linear equations with PARDISO," *J. Future Generat. Comput. Syst.*, vol. 20, no. 3, pp. 475–487, 2004.

[12] H. Kurose, D. Miyagi, N. Takahashi, N. Uchida, and K. Kawanaka, "3-D eddy current analysis of induction heating apparatus considering heat emission, heat conduction, and temperature dependence of magnetic characteristics," *IEEE Trans. Magn.*, vol. 45, no. 3, pp. 1847–1850, Mar. 2009.

[13] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, "Superfast multifrontal method for large structured linear systems of equations," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 3, pp. 1382–1411, 2009.

[14] O. Bíró, "Edge element formulations of eddy current problems," *Comput. Methods Appl. Mech. Eng.*, vol. 169, no. 3, pp. 391–405, 1999.

[15] T. Mifune, T. Iwashita, and M. Shimasaki, "A fast solver for FEM analyses using the parallelized algebraic multigrid method," *IEEE Trans. Magn.*, vol. 38, no. 2, pp. 369–372, Mar. 2002.